Scope Measurement on Large IT Projects in Texas: A Position Paper

By Herb Krasner, Don Shafer and Linda Shafer, February 2, 2018

Executive Summary

The new Section 2054.159 of the Texas Government Code (HB 3275, 85R) adds specific monitoring requirements for Major Information Resources Projects (MIRPs[10]) in Texas state agencies. It requires regular monitoring and reporting on project performance indicators of: schedule, cost, scope, and quality. **IT scope measurement is the focus of this report.**

The application of IT scope metrics serves the following basic purposes:

- To help achieve project goals and objectives
- To help define and deliver the right product/system
- To assist in estimation, planning and executing the project effectively
- To help release a quality product/system at the right time
- To help analyze and evaluate created work products (intermediate and final)
- Facilitate organizational improvement

Along with measuring the other MIRP performance indicators (cost, schedule, quality) [1], a successful IT scope measurement program in an organization will mean that:

- The measurement program results are actively used in IT decision making
- These measurements are communicated and accepted outside of the IT organization as well as within the project
- The measurement program is self sustaining
- Ultimately it will mean that a larger percentage of MIRPs will be considered fully successful

In this paper we recommend several specific IT scope metrics that should be used on all MIRPs.

- Balanced Scorecard (BAL),
- System/project size and its growth or change,
- System requirements metrics for: anomalies, quality, volatility, change impact and satisfaction.

How these are combined to answer the larger question of *are we on track* is discussed, as well as, how these metrics are used to address the typical scope challenges found on MIRPs. We conclude with a discussion of the next steps that the state needs to take to properly implement these recommended metrics.

It will be important for agency CIO's and IT Leaders (e.g. IRMs, IT PMOs, IT project leaders, vendor contract managers) to begin the process of defining more precisely what their IT scope objectives and performance indicator(s) would mean; and laying out the framework for collecting and reporting on all newly required MIRP performance measurements. It will also be important for state IT leadership to update the various policies, procedures, rules, handbooks, reporting instructions, and templates that govern the effective monitoring of MIRPs.

We are hopeful that the positions put forward in this paper provide useful guidance that can be adapted to the state, the agencies and local project needs as necessary. Please contact the authors listed in appendix 1 for more information or to answer questions about this and related reports.

Table of Contents

Context. The new Section 2054.159 of the Texas Government Code (HB 3275, 85R) adds specific measurement requirements for Major Information Resources Projects (MIRPs[10]) in Texas state agencies. It requires regular monitoring and reporting on project performance indicators of: *schedule, cost, scope, and quality.* Since IT project scope is a new area of required measurement for Texas MIRPs, particular attention will be needed to define the framework for effective measurement and derivation of reported scope performance indicators. Scope needs to be defined, evolved and controlled by effective management practices over the entire IT project lifecycle. In our previous position papers, we discussed the implications and challenges of implementing this new law for Texas state and agency IT leadership[11]; and also the more specific implications of measuring IT quality[1].

Introduction

Empirical research has shown that poor: scope management, requirements definition, and change management are the major causes of IT project failure[13]. Every year these failures cost huge sums of money. One of the major failure sub factors is the result of uncertain expansion or fluctuation in project scope, aka, scope change. Furthermore, scope change directly affects the project's budget, schedule, and finally the product's quality.

IT Project Scope Management includes those processes required to ensure that the project includes all the work required, and only the work required, to complete the project successfully. More precisely the term scope includes:

- **Product scope.** The features, functions, technology and quality attributes that characterize a product, service, and/or result; and
- **Project scope.** The work performed to deliver a product, service, and/or result with the defined product scope.

In many cases the term "project scope" is used to mean both of the above, as in the remainder of this paper unless indicated otherwise.

Product scope can be used to estimate project scope (i.e., schedule, budget, resources). Alternatively, constraints on project scope may be used to determine product scope. Constraints on both project and product scope may require tradeoffs among features, quality attributes, schedule, budget, resources, and technology.

Together project and product scope determine the effort needed to develop or modify an IT system. Effort is the primary cost driver for these projects.

There is a continuum of possible life cycles for IT system projects that lie within a spectrum from highly predictive life cycles to highly adaptive life cycles. Where necessary we describe the common elements of, and the distinctions between, scope management for predictive and adaptive life cycles, and where that may effect the measurements to be used for scope management.

A more quantitative approach to developing IT systems is highly desirable. To achieve this, the IT profession needs a set of useful measures that are understood and accepted by developers, users, and management. A practical, flexible scope measurement suite is thus needed, not only to

develop more reliable estimates, but also to better communicate with technical and nontechnical management on issues of scope, change, complexity, risk, and the like.

Establishing a reasonable set of scope metrics

Status, as defined by dictionary.com is "state or condition of affairs" at a certain point in time. What we need to track then is the condition of affairs of the scope dimension of our project at various points in time. The implication is we need to define the desired condition of affairs (our *objectives*) and *measure* against them over time to determine progress or lack thereof.

The Goal Question Metric Approach (GQM - 1984)[8] set out to identify the relevant metrics by defining specific goals for their measurement. These goals suggested the kinds of questions or models you may want to use, and these define the metrics you need. The models you select provide the framework for interpreting the metrics. For example:

Tactical Measure	Question Answered	Sample Indicator
Scope	Is our project scope stable/volatile/correct?	Number and Size of Change Requests
Time	How are we doing against the schedule?	Schedule Performance Index (SPI) = Earned Value ÷ Planned Value
Cost	How are we doing against the budget?	Cost Performance Index (CPI) = Earned Value ÷ Actual Cost
Resources	Are we within anticipated limits of staff-hours spent?	Number of hours overspent per phase or iteration
Quality	Are the quality problems being addressed?	Number of defects fixed per user acceptance test
Anomalies (issues, problems & defects)	Are we keeping up with our action item list for anomaly resolution?	Number of action items behind schedule for resolution, defect counts

Defining goals involves specifying the objects you are measuring, e.g. a product, process, a model, the focus of interest, e.g., cost, defect removal, change, reliability, user friendliness; the purpose, e.g., to characterize, analyze, evaluate, predict, the perspective of the person wanting the information, e.g., the manager, developer, organization; and the context, e.g., the organization's characteristics and context variables. All these help define what measures you need and how you are going to interpret them.

The Process of Defining Scope

It will be impossible to measure an undefined scope of work. Most large IT projects will follow a well defined, disciplined and systematic approach to establishing scope, using known best practices and tools. In general, scope should be crisp and well defined, and derived from the business goals. Identifying features and functions that are *specifically excluded* from work products and deliverables can be extremely useful, as it helps sets firm, clear boundaries, helping to curb scope creep and gold plating.

The documentation of the scope of the project will define the boundaries of the project, establish the responsibilities of the team, and set up procedures for how work that is completed will be verified and approved. This documentation is often referred to as the *scope statement* or *statement of work*, and would be augmented by a: business case analysis, project charter, project plan, and a project dictionary.

The scope definition also helps to manage stakeholder's expectations from the very beginning of the project. Therefore, a scope definition is considered "proper" only when it is acceptable to the stakeholders. In fact, the quality of the scope definition is one of the initial components that determine the fate of the project.

To fully define the scope of a project in the traditional mode, the following would be identified:

- Project objectives (within the broader goals of the organization)
- Requirements
- Deliverables
- System Development Lifecycle (SDLC) and its Sub-phases
- Tasks
- Resources
- Budget
- Schedule
- Quality objectives

Once these parameters are established, the limitations of the project can be clarified and the aspects that are not included in the project identified. By doing this, the project scope will make clear to all stakeholders what will and will not be expected in the final product and/or services.

For an ideal predictive life cycle IT system project, the initial project and product scope statement is meant to be a static document, however **this is rarely the case in practice**. In an adaptive life cycle IT system project, the scope statement is a list of high-level themes, epics and/or functional objectives. This is planned to be an evolving document that is bounded by overall project constraints. Systematic evolution of scope is an important factor in both adaptive life cycles and predictive life cycles. The mechanisms for that evolution are quite different however.

The typical problems that arise when defining and documenting Project Scope are:

- **Ambiguity:** Ambiguity in scope often leads to unnecessary work and confusion.
- **Incomplete definition:** Incomplete scope definition leads to schedule slips which lead to cost overruns.
- **Transience:** Transient scope leads to scope creep—the primary cause of late deliveries and "never-ending" projects.
- **Uncollaborative scope:** A scope definition that is not collaboratively prepared causes misinterpretations in requirements and design.

These can lead to execution problems once the project has begun, and therefore early and rigorous continued scope review is necessary.

A process for controlling IT project scope in a traditional project is defined by the PMI PMBOK standard, Chapter 5 and the Software Extension to the PMBOK, Chapter 5. We assume that the reader is familiar with that approach[9] and terminology for the remainder of this paper.

Measurement Implications for Scope Management

It is assumed by this paper that the given MIRP is already practicing earned value management (EVM[11]) in the areas of cost and schedule control. EVM metrics (e.g. Estimate To Complete (ETC)) will contribute to scope control by using established baselines to control underestimation of the complexity of the problem and/or the emerging solution system, and by looking at planned versus actual accomplishments. This helps by illuminating the project from the perspective of cost and schedule overruns. The measurement of earned value was covered in our initial position paper[1]. Earned value may be determined differently for adaptive (e.g. agile) projects.

Three additional scope metrics categories are recommended here: balanced scorecard, system/project size and requirements metrics.

(i) Balanced scorecard is intended to improve the development of a system by determining the contextual parameters related to financial, customer, internal process development, and learning/growth perspectives.

(ii) System size attempts to measure the inherent size of the emerging system from requirements all the way through code and operation. The relative size of the project attempts to categorize the project in order to apply previous historical data and lessons learned.

(iii) Requirements metrics focus on tracking and documenting the changes to the product requirements and then relating those changes to the actual needs of the project. In addition, requirements metrics track and document the changes along with their causes and the entity responsible for the changes.

With these basic facts, it can be said that, with the proper preparation of management plans at the beginning of the project, along with an implementation of suitable IT measurements, are supporting factors for scope management in any large IT system project.

Balanced scorecard[2] (BAL)

BAL is a management tool that is used by an organization to monitor their performance against strategic goals and project objectives. The term "scorecard" signifies quantified performance measures and "balanced" signifies the system is balanced between short-term and long term goals, objectives, financial and non-financial measures, lagging and leading indicators and internal and external performance perspectives.

Metric 0: BAL.

BAL maps the organization's project goals into performance metrics in four areas: financial, customer relationships, internal processes, and learning/growth. These perspectives provide relevant feedback as to how well the project plans are being executed so that adjustments can be made as necessary. The four perspectives are:

(i) **Financial**. Financial performance measures indicate whether the project's implementation and execution are contributing to bottom- line improvements. The main financial objectives for an IT project are typically: *ROI, value achieved, productivity improvement, cost of ownership and/or cost reduction.* These are documented in the business case prepared prior to project initiation, and are then tracked during project execution to ensure that agreed upon value is in fact delivered. The primary metric for this perspective is usually Return On Investment (ROI).

(ii) **Customer**. Customer perspective mainly focuses on the measurement of customer needs and fulfilling their expectations. Customer satisfaction metrics are created and used. Customer satisfaction can be determined by conducting surveys among the customers, or by means of other feedback mechanisms (e.g. user alpha and beta testing). SLA achievement metrics are used for IT-as-a-service situations.

(iii) **Internal process**. The internal process perspective based metrics begin with the team's understanding about the effectiveness of the definition, development, delivery and sustainment process (i.e. SDLC or Lifecycle). The conventional performance measurement system focuses on monitoring and improving existing processes, as well as, identifying entirely new processes at which a project must excel to meet customer needs and project objectives. Metrics include: *process maturity, process effectiveness, average cycle time, etc.*

(iv) **Learning and growth**. The learning and growth perspective includes employee skills development and organizational cultural attitudes related to individual, team and organizational self-improvement. In the current climate of rapid technological change, it is necessary for knowledge workers to be in a continuous learning mode. Learning and growth metrics guide managers in focusing training funds where they can help, and lead to the need to invest in the three categories of learning and growth such as employee capabilities; information systems capabilities; and motivation, empowerment, and alignment. Metrics include: *staff turnover rates, learning outcomes, team effectiveness, etc.*

IT project specific metrics can be determined in each of these four categories in accordance with the specific situation. Good examples of using this technique in the software development industry[6,7] reveals the potential for use in all MIRP situations.

Project Size

Metric 1: Total Effort and its growth rate

In **project** terms, "**size**" is usually an all encompassing designation, used to quantify the overall "extent" of the **project** effort, usually accounting for duration, cost, complexity, staffing requirements and related parameters. This allows IT projects to be characterized in comparative terms. We prefer the rough order classification scheme of: tiny, small, medium, large, extra large, and mega. The relative budget ranges for this scheme are ~: <$100K, <$1M, <$10M, <$100M, <$B, and >$1B. The approximate total budget size can be computed by the total number of person-hours required times the average hourly rate of all members contributing to the project.

Projects of different sizes may need to operate quite differently depending upon their objectives, and therefore the formality of their scope control processes may be different.

However, the total number of work hours is a simple, meaningful metric for the size of a project's effort. Hours consumed relative to plans are easily tracked with a time recording tool. The effort growth rate should follow a predictable pattern (e.g. a *Raleigh-Norden* curve). Another possible measurement of project size is for example, the number of activities/tasks in the WBS or task lists.

System Size

IT system size is the main input parameter to effort estimation models. However, there is no standard, single measurement for IT system size; unlike other industries (e.g. that use weight – as in the food industry and square feet for building footprint). Each sizing method has its advantages and disadvantages and professional advice should be sought before choosing the method that best suits your project. Benchmarking is facilitated by using industry data, such as found in the ISBSG Repository[12] or (preferably) the organization's own historical project database.

Metric 2: Functional size and its growth.

IT functional sizing is used to estimate the size of a system, application or component to support cost estimating, progress tracking, and other project management activities.

The functional size attribute of an IT system depends upon how the requirements are defined. Commonly used early metrics are the number of pages or lines of text or the countable number of requirements (perhaps weighted), in the requirements definition document or product backlog. There are several system size metrics to consider, depending upon the type of development project and the lifecycle model chosen. These include counting, for example: function points, non-functional points, enhancement points, feature points, story points, number of testable requirement statements, epic/thematic specifications, use cases, user screens, database size, reports produced, weighted system modules/components, source lines of code, number of lexical tokens, number of test cases, documentation number of pages/words, etc. ***All large IT projects wrestle with this concept of measuring the inherent size of their emerging system throughout the lifecycle.***

An appropriate blend of these measures should be defined and tracked for each specific project. All of these different kinds of size metrics can convert to the more common metrics of *work product object counts* and ultimately *effort*. The most important thing is to monitor the growth pattern for these size measures, and look for anomalies relative to expectations in that pattern.

The older method of counting lines of code is often too late and less useful for today's component-based systems. International Function Point Users Group (IFPUG) function points is currently the most commonly used IT sizing metric. It is better suited to applications that contain user interfaces, reports, system functions and database tables.

Basically, a number of function points are assigned to a screen, depending on the number of data elements on the screen/report and the number of database tables that are referenced. Function points are also assigned to tables depending on their size. The total is the unadjusted function point count (UFP). There is then an adjustment step to cater for the other attributes that need to be considered when developing a system. There are 14 attributes to assess. For details on IFPUG function point counting go to IFPUG. There is also an emerging standard for automating the measurement of an IT system size using function points. For more information see http://it-cisq.org/standards/software-sizing-standards/

On agile projects the most common size measurement is story points, and the number of story points delivered in an increment (e.g. sprint) is assumed to be relatively stable in order to achieve consistent velocity (flow). There is a known relationship between story points and function points as a measure of IT system size, which can be empirically developed for each agile team. However this conversion approach (backfiring) is currently an emerging practice. It is advised that agile teams use both story points and function points to measure size over the development lifecycle.

Requirement metrics

In common usage, a **requirement** is something that is wanted, needed, asked for, or demanded. Broadly a "requirement" is defined as a condition or capability that is necessary for a system to meet its objectives. For IT projects, the term *requirements* is defined within project objectives and usually means specific capabilities that a system must provide in order to solve a problem.

An IT system requirement is a statement of one of the following:

1. What a system must do (functional, *shall*)
2. A known limitation or constraint on resources or design (e.g. capacity or speed)
3. How well the system must do what it does (e.g. quality attributes)

The collection of all such statements is called the requirements definition set or suite or specification. We would like each requirement and the entire suite to have known desirable properties (e.g. clarity, completeness, consistency, etc.).

Requirements include functional requirements; performance requirements; and requirements for hardware, software, and user interfaces. They may also specify development or quality assurance standards for both product and project. They can be stated either quantitatively or qualitatively, but the preference is to quantify whatever possible[4].

Initially, in predictive life cycle projects, there is an attempt to develop a set of system requirements that are complete, correct, consistent, and appropriately detailed. These requirements provide the basis for determining the evolving scope of the project and for developing the Work Breakdown Structure (WBS) and the work packages. Project scope is then managed by controlling changes to the requirements and the work activities needed to implement those requirements.

The impact of changes to product requirements on the project schedule, budget, resources, and technology may require revision of the project scope and is reflected in changes to the plans,

WBS or task list. Change control boards and a version control system are typically utilized in predictive projects to manage the changing scope of an IT system project.

Since all of the requirements identified in upfront elicitation will not usually be included in the product, defining scope involves choosing the requirements that will be part of the product scope. This issue is commonly dealt with by prioritizing the requirements using criteria that include the wants and needs of the customer and user communities, and the value added by each requirement. Risks, assumptions, and constraints are also taken into consideration when defining scope.

IT system requirements for predictive life cycle projects are typically documented in an online repository of baselined requirements. Requirements for future iterations of an adaptive life cycle are maintained in product feature backlogs, candidate feature lists, story lists, or in a more automated requirements management tool.

Requirements documentation, including traceability, is particularly important because of the intangible nature of an IT system. A requirements traceability matrix provides visibility from requirements to intermediate work products (e.g., design documentation, test plans, test results), and then to the components of the deliverable product.

There exist techniques, methods and tools to help business analysts, product owners, managers and subject matter experts create, clarify, and confirm business, stakeholder, solution, and transition requirements for successful IT project outcomes.

The metrics that are useful in identifying the risks of a project by identifying errors and changes in the requirements definition suite are known as requirement metrics. These metrics validate the documented requirements against actual requirements. They evaluate whether the requirements are of good quality or not. A single metric cannot ensure overall requirements quality; therefore, multiple metrics should be used in this area.

Requirements Objective: Deliver a system with the agreed upon functionality and required non-functional attributes. The following 5 requirements metrics support that objective.

Metric 3: Scope anomalies, problems and deficiencies

An anomaly is any condition that departs from the expected [Institute for Electrical and Electronic Engineers (IEEE) Standard 1044]. The expectation that establishes the context for an anomaly may come from any number of sources (e.g. requirements documentation, user perceptions). An anomaly is not necessarily a problem or a defect; but could generate one or more. Significant anomalies should be recorded and thus require management attention.

A scope problem is influenced by the SDLC chosen. What is a problem in a waterfall project is an *opportunity* in an agile project. In any case, an identified problem requires action to rectify. There are many kinds of IT scope problems. For example, due to:

- Lack of domain knowledge
- Lack of scope definition/elicitation expertise
- Too much/too big scope
- Lack of prioritization

- Undocumented scope
- Vague scope
- Lack of measurability in scope
- Frequent changes in scope

IT scope problems are usually tracked in a Problem Reporting tool, and are managed to closure in a disciplined fashion.

A scope deficiency (i.e. defect) is determined in relation to a defined requirements quality model (either explicit or implicit). Such defects are tracked in a defect management tool. Sadly, too many organizations and projects do not report and track requirements defects.

Metric 4: Requirements quality

The purpose of having *requirements* is to form a common understanding and agreement between the stakeholders and the project team about delivering a system with specific functionality to their customers. This should include relative priorities and valuations. Historically this area has been the most problematic for larger IT projects[1].

This area varies greatly with the level of desired specificity of defined system objectives. Committed requirements/functions/features/use cases/user stories versus delivered results meeting defined "doneness" criteria are usually tracked. The term SMART is an acronym for Specific, Measurable, Achievable, Realistic and Time-bound. SMART is a much-used tool in determining if requirements are well written. Completeness and consistency are also important. Many measure *volatility* as a quality indicator.

In the conventional, waterfall model of development a standard like IEEE Standard 830 could be applied. That model requires a Software Requirements Specification to be correct, unambiguous, complete, consistent, ranked for importance, verifiable, modifiable, and traceable. Every one of these qualities could be (and should be) verified in their own specific ways. For example, ambiguity detection in English language requirement statements is challenging but doable. This view of requirements quality rarely applies to the innovative; agile IT projects typically done in IT shops today.

Content related requirements quality (as opposed to syntactical views) refers to goal models that connect expected content to project success factors. Such models need subject matter expert interpretation. These models can lead to a set of quality metrics to facilitate achievement.

Agile teams use a hierarchical model (aka roadmap) of themes, epics, and product backlogs of user stories to manage their requirements. Product owners prioritize the user stories and help create acceptance criteria. To enable delivering products with sufficient quality agile teams need to have user stories that are ready at the start of each sprint. Teams can use a Definition of Quality (DoQ) to check the user stories. A DoQ states the criteria that a user story should meet to be accepted into the start of an iteration. For example, using a model such as INVEST to grade the individual stories in six factor areas.

I – Independent

N – Negotiable
V – Valuable
E – Estimate-able
S – Small
T – Testable

Writing quality user stories is starting to emerge into a best practice, as we are now moving into the 2nd generation of disciplined agile methods.

Emerging research has started to define the contents of quality users' stories. This includes, for example, a set of criteria for user story quality. This includes *syntactic* (textual) criteria of: well formed, atomic, and minimal. This also includes the *semantic* (meaning) criteria of: conceptually sound, problem-oriented, unambiguous, and conflict free. This also includes the *pragmatic* (subjective interpretation) criteria of: full sentences, estimatable, unique, uniform, independent and complete. Tools are starting to emerge for analyzing user stories for these quality criteria.

Metric 5: Requirements volatility

Gause and Weinburg[5] explain that false assumptions not corrected in the requirements definition phase may cost 40 to 1000 times more to remedy after the system has already been fielded. For this reason, the magnitude of changes in the project requirements are considered as one of the leading indicators of potential problems.

The degree of measuring the requirement changes over a time period is called requirements volatility. It also is used to help determine the reasons of requirement changes. These factors are measured to know whether the changes are consistent with the current set of development activities or not. It helps in tracing future requirements, design, and code volatility by indicating the requirements changes.

The best metric for requirements volatility is to determine what requirements have changed between baselines. A requirement may have changed multiple times between baselines, but should be recorded as a single change. In addition, a new requirement that may have undergone multiple editorial changes should be treated as a new requirement, until a new baseline is struck. Measuring volatility this way will identify the volatility of existing requirements, the number of new requirements; and you can easily determine the number of deleted requirements between baselines. These measures depend on the project striking baselines during key milestones. Requirements management tools (e.g. DOORS) along with Excel spreadsheets can provide the visualization of requirements volatility over time.

The requirements volatility metric supports the measuring of the number of requirements added, deleted and modified. It also determines the causes of adding, deleting and modifying the requirements and classifying those requirements with the reason for change. A proper implementation of the requirements volatility metric has the following advantages:

- Determines the number of initially allocated requirements. The requirements volatility metric measures the number of initially allocated requirements. It includes all the technical and non-technical requirements provided by the customer. This metric describes the level of requirements volatility along with the number of final allocated requirements as well as the number of changes allocated per requirement.

- Determines the final requirements. This metric measures the number of allocated final requirements. It also includes all technical and non-technical requirements to build the final IT system products.
- Tracks the number of changes per requirement. This metric tracks the number of changes made to each requirement. Along with describing the level of volatility of the requirements, this metric also describes the impact of changing the requirements in the IT system process.
- Tracks the number of changes in a specific time period. This metric contains the number of changes of requirements for a specific time period, such as a week or a month. It describes the degree of volatility of the requirements. Its value should decrease towards the end of the IT system lifecycle (indicating convergence of requirements). It is measured during the project lifecycle.
- Causes of change. This metric collects all the causes of requirement changes and categorizes them. It helps in identifying the most common causes of change in the IT system process and can be used to improve the IT system process.
- Identifies who requested the change. This metric helps in identifying the source of the change, the reason for implementing a specific functionality, and anticipates the source of changes in the future.

Due to these specific functionalities of the requirements volatility metric, it is suitable to determine the causes of scope changes and prevent project scope creep.

In agile projects, by committing to only what needs to be done in the next incremental delivery, the overall stability of requirements converges, thus resulting in a higher quality product delivered to customers. Even in this case baselines can be established and volatility tracked as the backlog is scrubbed during iteration planning and then iteration closure.

The main purposes of this aspect are to: track progress or lack thereof, enable corrective actions when necessary, and to detect variance from our plans. Periodic status meetings/reports, and milestone reviews are based in factual data.

Metric 6: Impact due to functionality change requests.

Metric: Variance in project plan due to functionality change requests. The measurement mechanism for scope is project change requests, and the impact of that change to the project. This should be measured at least monthly and in some situations weekly.

Change in functionality can be observed by the number of change requests related to functionality (or quality). As well, having requirements completion running late could mean that there is difficulty obtaining requirements and, thus, defining functionality. Measuring the variation in time between target sign-off date for requirements and the actual sign off date can be a leading indicator that there may be scope challenges.

A defined scope baseline agreed to by the customer forms the functional baseline for the entire project planning and development. Any change to defined scope should happen in controlled manner. So an important metric to track i.e. the number of change requests coming from customer for the already baselined scope of work. Each and every change request, once approved by an internal change control board (CCB), requires update to Scope baseline which in turn has a cascade impact on cost baselines and schedule baselines and resource plans.

Uncontrolled change requests often result in project scope creep and further impact negatively on the project cost/schedule, which is the worst thing to happen for any project. Based on the magnitude of the variance from original scope baseline, CCB should decide whether to accept or reject a change request and this decision should be communicated back to the customer. In the case of acceptance of a change request, the impact on project cost, schedule and quality should be clearly communicated in written form to the customer and a written agreement from the customer is secured before proceeding. The formality of this process is often relaxed in agile projects.

One metric for monitoring scope is the frequency and magnitude of project change requests. This includes vendor contract amendments if part of the work is sourced. In most organizations, change requests can be submitted at any time. However, there may be regularly scheduled change request review meetings by CCB or its equivalent after which the status can be updated. In agile environments scope is allowed to change at identified activity points (e.g. backlog grooming after iteration closure), but still is tracked for impact. The trend period should be aligned with these meetings/reviews to capture the change in information.

A change in scope can be observed by the number and size of change requests related to functionality. Measuring the variation in time between target sign-off date for requirements and the actual sign off date can be a leading indicator that there may be scope challenges.

Metric 7: Requirements satisfaction

Requirements satisfaction is a stand in for customer/user satisfaction prior to the existence of demonstrable functionality. As such it relies on the discipline of requirements change management to keep track of the current baseline, and the traceability mapping between the baseline and the current set of work products.

Many teams find it difficult to answer the following questions as the project proceeds through its SDLC phases:

- Do the essential requirements meet the basic needs of the intended users?
- Does the system architecture support and enable all of the requirements?
- Do the detailed designs incorporate the architectural constraints and address the allocated requirements?
- Does the code satisfy all of the above and efficiently implement the requirements?
- Do the test scripts, test cases and test environment assure us that the system will work properly in its intended environment?
- Have we sufficiently tested the system to ensure that it satisfies all requirements?
- How and when will I know that we are done with this project?

Agile teams may ask a slightly different set of questions, but eventually these must all line up.

In every project there must be a clear understanding of each requirement and accountability or ownership of each requirement. The team uses the requirements documentation and traceability matrices to establish understanding and ownership of each requirement and to track the completion of each requirement into the new system as implemented. Requirements traceability is assumed in order to measure requirements satisfaction at intermediate milestones and

checkpoints. Requirements traceability can be thought of as a process of documenting the connection and relationships between the initial requirements of the project, the intermediate work products and the final product or service produced. This discipline is assumed to be active during all project phases. The technique is considered as a bilateral approach in that it tracks the requirements forward by examining the output of the deliverables and also backwards by looking at the business requirements that was specified for that feature of the product. A traceability tool is essential.

Requirements satisfaction is therefore conceived as the completeness of the mapping of the current state of the project (and its work products) back to the current baseline set of requirements. Either that mapping is *complete, partially complete or unable to be determined*. The latter result is undesirable.

At the delivery point(s) of the project, customer satisfaction survey data should be collected, analyzed and reported for this metric.

Conclusions

What about Overall Status of the project being tracked?

The overall objective for an IT project is: *Deliver a system/product that meets an expected level of quality within the stakeholder agreed upon functionality and constraints.*

The two most common overall status-rating techniques are Worst Child and Weighted Roll-up. We prefer the latter of these two.

Worst child: In this situation, the overall status takes on the value of the worst child (child = sub factors of success objectives) status. For instance, if the budget status is yellow and all others are green, the overall status would be yellow. This approach drives attention to pushing projects to be "all green". It is the most thorough approach, however it may be difficult to distinguish relative priorities between projects. For example, a project with only one child status of red and a project with all children status red will both have and an overall status of red.

Weighted roll-up: In this case, each sub-status is provided a relative weighting of its importance. For example, you could place greater relative importance on budget over scope, schedule and quality. Of course, you may have more thresholds than the traditional red, yellow and green. The three thresholds are used here as a simple example.

Measuring overall status is tricky because, suddenly, you may have lost sight of the context of the status. If the overall status is 'yellow' it is not clear what needs to be addressed. Rather it is a sign that the project needs attention. Overall status would be a logical combination of the other status items.

If a project is off track as indicated by the status, management needs to make some decisions and take action to correct the problems associated with the status. Once the action has been executed management needs to know if it is making a difference. To do this we need to capture and report on the status *trend*. The trend is simply how the status is changing from one reporting period to

another. The principal consideration is the level of granularity of reporting period. Multiple levels of reporting periods (and thus multiple trends) may be required.

In a controlled environment, scope only changes through Change Requests. In most organizations, change requests can be submitted at any time. However, there may be regularly scheduled change request review meetings after which the status is updated. The trend period should be aligned with these meetings in order to capture the change in information.

It is natural for parts of a large IT project to change along the way (after all, projects are *all about* change – from the way things were in the past, to the way they will be in the future). So, the better the project has been "scoped" at the beginning, the better will be the baseline to manage change against. How changing scope is managed (and measured) is therefore a prime determinant of project success. This is especially true as organizations move from adhoc projects, to disciplined and mature projects, and then towards more adaptive agile and lean projects.

IT scope metrics were recommended here to help achieve more successful MIRP performances. Assuming that some form of EVM is already practiced, these are:

1. BAL
2. System size and its growth
3. Requirements metrics, including:
    a. Change request impact
    b. Scope anomalies, problems and deficiencies
    c. Requirements quality
    d. Requirements volatility
    e. Requirements satisfaction

A proper implementation of BAL improves tactical project performance in the context of strategic goals. The final metrics that play an important role in managing scope are requirements metrics.

These metrics properly track the requirements and document any change that occurs in the requirements along with the factors that cause the changes. Apart from these, the metrics have the capabilities to relate the changes with the needs of the project, which support the project management team to determine the necessity of those changes, deletions and modifications.

As requirement volatility can be high in the initial phase of IT system development, it should be reduced as the project progresses so that further development is not affected.

The Challenging factors to effective IT project scope management as related to our recommended metrics are shown in the table below. These underlying causes of these challenges may emerge differently in different kinds of projects.

Table 1: Recommendations for Scope Measurement Vs. common scope problems

No.	Scope challenges	Recommended metrics and plan

1	Ambiguous project scope	BAL, Scope baseline quality, Size
2	Insufficient resource allocation	BAL, Requirement metrics, Size
3	Lack of end-user involvement	BAL, Requirement metrics
4	Underestimating the complexity of problem	EVM, Scope baseline quality, size
5	Ineffective communication	BAL, Requirement metrics
6	Vague and incomplete requirements	Requirement metrics, Requirement management plan
7	Change in customer needs	Requirement metrics, Requirement management plan
8.	Change in environment, technology and/or organization structure	BAL, Configuration management plan, Size
9	Platform changes	BAL, Configuration management plan
10	Lack of change control management	Change control management plan, Requirement metrics and Requirement management plan
11	Addition of extra features	Change control management, BAL, Size and Requirement metrics

Next Steps

Managing a MIRP for success is challenging. Effective measurements can help; especially in the area of IT scope control. We assert that IT scope must be defined and measured if significant improvement is to be achieved.

It will be important for state agency CIO's and IT Leaders (e.g. IRMs, IT PMOs, IT project leaders, vendor contract leaders) to begin the process of defining more precisely what their IT scope objectives and performance indicator(s) would mean; and laying out the framework for collecting and reporting on all required MIRP performance measurements.

In many leading edge organizations, it is hard to imagine managing IT development and evolution without metrics to help guide decision-making. There are many practical uses of IT measurement, but four areas in particular deserve our attention:

1. Project estimation, progress monitoring and corrective action
2. Evaluation of work products
3. Discipline-driven process improvements
4. Experimental validation of best practices

We are hopeful that the positions put forward in this paper provide useful guidelines that can be adapted to the state, the agencies and local project needs as necessary. To the extent that other states are following a similar path to effective measurement of large IT projects, perhaps the guidance contained herein may help them as well.

References

1. IT Quality: Measurement Implications for Large IT Projects in Texas
   By Herb Krasner, Don Shafer and Linda Shafer, November 8, 2017
   http://it-cisq.org/wp-content/uploads/2017/11/IT-Quality-Measurement-Implications-for-Large-IT-Projects-in-Texas-Nov-2017.pdf
2. R.S. Kaplan, D.P. Norton, *The Balanced Scorecard. Translating Strategy into Action* (Harvard Business School Press, 1996)
3. N. Fenton, *Software metrics: A Rigorous Approach* (Thompson Computer Press, 1995)
4. Krasner, H. (1989) REQUIREMENTS DYNAMICS IN LARGE SOFTWARE PROJECTS: A Perspective on New Directions in the Software Engineering Process, In the Proceedings of the 11th World Computer Congress (IFIP89), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, August, 1989
5. D.C.Gause and G.M.Weinberg, Exploring Requirements: Quality before Design, Dorsethouse Pub. New York, NY, 1989.
6. Balanced Scorecard Framework In Software Project Monitoring , Journal Of Engineering Management And Competitiveness (JEMC) Vol. 1, No. 1/2, 2011, 51-56 ISSN 2217-8147
7. Scorecard And KPIs For Monitoring Software Factories Effectiveness In The Financial Sector, International Journal of Information Systems and Project Management, Vol. 1, No. 3, 2013, 29-43, ISSN (print):2182-7796
8. *The Goal/Question/Metric Methodology: a practical guide for quality improvement in software development*, Solingen and Berghout, 1999, McGraw-Hill Publisher
9. *A Guide to the Project Management Body of Knowledge*, 6th Edition; and the *Software Extension to the PMBOK® Guide (Chapter 5)*. http://www.pmi.org/pmbok-guide-standards
10. MIRP Definition, A MIRP in Texas is typically > $1M and/or longer than a year to complete, or involving more than 1 agency. See Texas Government Code, Chapter 2054.003 for details.
11. Measurement Implications for Large IT Projects in Texas, By Herb Krasner, August, 2017
    http://it-cisq.org/measuring-it-project-performances-in-texas-house-bill-hb-3275-implications/
12. ISBSG repository, http://isbsg.org/
13. Taylor, A. (2000), IT Projects: Sink or Swim?,
    http://archive.bcs.org/bulletin/jan00/article1.htm

Appendix 1: Authors' Contact Information

Primary author: Herb Krasner
Email: hkrasner@utexas.edu
Phone #: 830-693-0631
Bio: http://www.ece.utexas.edu/people/faculty/herb-krasner

Author: Don Shafer
Email: dshafer@athensgroup.com
Phone #: +1 512.663.1459
Bio: https://www.linkedin.com/in/donshafer/

Author: Linda Shafer
Email: lshafer@computer.org
Phone #: +1 512.474.4398
Bio: https://www.linkedin.com/in/linda-shafer-15a0218/

